Empir Software Eng (2013) 18:139–176
DOI 10.1007/s10664-012-9198-8

A study of methods for textual satisfaction assessment

Elizabeth Ashlee Holbrook · Jane Huffman Hayes · Alex Dekhtyar · Wenbin Li

Published online: 16 February 2012

Editor: Daniela Damian

Abstract Software projects requiring satisfaction assessment are often large scale systems containing hundreds of requirements and design elements. These projects may exist within a high assurance domain where human lives and millions of dollars are at stake. Satisfaction assessment can help identify unsatisfied requirements early in the software development lifecycle, when issues can be corrected with less impact and lower cost. Manual satisfaction assessment is expensive both in terms of human effort and project cost. Automated satisfaction assessment assists requirements analysts during the satisfaction assessment process to more quickly determine satisfied requirements and to reduce the satisfaction assessment search space. This paper introduces two new automated satisfaction assessment techniques and empirically demonstrates their effectiveness, as well as validates two previously existing automated satisfaction assessment techniques. Validation shows that automatically generated satisfaction assessments have high accuracy, thus reducing the workload of the analyst in the satisfaction assessment process.

Keywords Methods for SQA and V&V · Requirements/specifications · Validation · Tracing

E. A. Holbrook
Lexmark, Lexington, KY 40508, USA
e-mail: ashleeh@gmail.com

J. H. Hayes (✉) · W. Li
Department of Computer Science, University of Kentucky, Lexington, KY 40506-0495, USA
e-mail: hayes@cs.uky.edu

W. Li
e-mail: wenbin.li@uky.edu

A. Dekhtyar
Department of Computer Science, California Polytechnic State University, San Luis Obispo, CA 93407, USA
e-mail: dekhtyar@calpoly.edu

1 Introduction

Mission- or safety-critical systems often undergo Independent Verification & Validation (IV&V). One of the most important questions that the IV&V agent helps answer is: "does this system meet the requirements?" For example, for an instrumentation and control subsystem of a nuclear power plant, the developer must build and present a safety case to "prove" that the system meets all safety requirements. In fact, this activity takes place throughout the lifecycle, not just at the end when code is delivered. At the end of the design phase, for example, the developer building the safety case has to demonstrate that the design specifications satisfy all safety requirements. To do this, the developer must first determine the traceability relationship, known as a requirements traceability matrix (RTM), between the safety requirements and the design elements. The developer will use tracing procedures to find relations between the textual safety requirements and the textual design elements. The constructed RTM shows the relationships between the safety requirements and the design elements *currently present in the documents*. Next, the developer asks the question: "have all safety requirements been met?" To answer, (s)he will check if the RTM encompasses every relationship *that should be present*. The activity that answers the latter question is called *satisfaction assessment*. To perform satisfaction assessment, the developer will examine each safety requirement and the design elements traced to it in the RTM and make a judgment on whether or not *every aspect* of the safety requirement is satisfied by some design element or collection of design elements.[1]

Satisfaction assessment is typically performed manually. The steps involved are:

1. Read a high level element (a safety requirement in this case).
2. Read each low level element (design elements in this case).
3. Determine whether each aspect of the high level element has been addressed by the low level elements.
4. Assign a label to the high level element (satisfied, partially satisfied, or not satisfied).
5. In the case of "partially satisfied" high level elements, document the portions of the high level element that have not been addressed.

Currently there are no tools to automate or semi-automate this process. Our work provides a degree of automation for step three, supports step five, and allows the IV&V agent (or developer) to perform other steps within a single software tool.

Before examining the process of satisfaction assessment, we define key concepts. A ***software requirement*** is a statement of what a software system must be capable of doing. Software requirements are typically one sentence in length and may include a second or third sentence that defines terms in the main sentence. Requirements, formatting variations aside, usually include three pieces of information: a) a subject; b) a modal verb, typically "shall;" and c) an object phrase describing what is required. The subject of a requirement is typically the name of a software system or a component contained within it. The modal verb may be "will," "shall," or another word. Finally, the requirement sentence must contain an object phrase. This describes an attribute or behavior of the subject of the sentence. For example, consider the requirement, "*The system shall write files to .PDF format.*" "The system" is the subject, "shall" is the modal verb, and "*write files to .PDF format*" is what the subject must be able to do. Requirements may be longer, containing additional sentences that provide

[1] The above scenario examines the case of a developer, but in many mission- or safety-critical projects, an IV&V agent is responsible for performing satisfaction assessment.

more detail in free form. For the given example, a second requirement sentence may describe .PDF format, or may say where output files are to be stored (Holbrook 2009).

A ***software design element*** has a less rigid format than a software requirement. Textual design elements describe how a requirement will be implemented or describe a requirement in more detail. For example, a software design element may be:

> "The .PDF format used by the system must follow standard format. The .PDF files will be written to a folder named 'output' that is located inside a user's home directory within the file structure. A .PDF reader will also be included in the system software package" (Holbrook 2009).

Requirements are matched to design elements via requirements tracing (Gotel and Finkelstein 1996), which can occur either when creating the design document or after the fact. Traceability links between requirements and design elements (or elements from other software artifacts) capture a range of possible relationships. In the past, work in requirements tracing has helped analysts determine whether requirements are related to design documents (Gotel and Finkelstein 1996; Antoniol 2002; Cleland-Huang et al. 2005; Hayes et al. 2006a, b; Marcus et al. 2005; Spanoudakis et al. 2003). Since the tracing process is long and arduous for analysts, recent research has concentrated on finding ways to automate the tracing process (Antoniol 2002; Cleland-Huang et al. 2005; Hayes et al. 2006a, b; Marcus et al. 2005). Automated methods analyze the textual artifacts, such as requirements and design documents, broken into individual elements and produce ***candidate Requirements Traceability Matrices*** (RTMs): collections of *suspected* related requirement/design element pairs (Asplaugh and Antón 2008; Hayes et al. 2006a, b; Di Lucca et al. 2001; Cuddeback et al. 2010; Gotel and Finkelstein 1996; Hayes et al. 2003; ISO9000:2000; Marcus et al. 2005; Spanoudakis and Zisman 2005; Spanoudakis et al. 2004).

The same automated techniques that are used to *generate* an RTM can be used to determine which aspects of a requirement have been addressed by low level elements and which have not. Nonetheless, satisfaction assessment as an activity is different from requirements tracing. To borrow an analogy from theory of computing, the tracing process establishes the correctness (soundness) of the trace relationship, whereas the satisfaction assessment process establishes its completeness.

In this paper, we investigate *automated methods for satisfaction assessment*. The formal definition of satisfaction assessment is presented in Section 2 of the paper. Informally, satisfaction assessment establishes the quality and/or completeness of a discovered traceability mapping between a pair of artifacts. Rather than concentrating on which design elements relate to a specific requirement, satisfaction assessment investigates *whether all the aspects of the requirement are addressed in the design elements that trace to it*. A requirement and design element may be related, *but the design element may not fully address the nature of the requirement*. For example, consider a hypothetical user management system for an online banking system. Suppose the system has a requirement "*At least three users must be able to log into the system simultaneously after providing proper login credentials*" and a design element "*user records in the system will contain the following attributes: userId, password, name, age, address, and account number.*" These two elements may be related in an RTM – both involve users and their «login credentials». However, the design element does not fully satisfy the requirement, as the design element contains no explanation of how the system would allow multiple users to log in at the same time.

In small-scale projects, it is possible to manually validate whether requirements have been fully met by design. For each requirement, analysts can read the project design specification, searching for and highlighting portions of individual design elements that

satisfy the requirement. However, many software projects are very large, having several hundred requirements and design elements. Unfortunately, the scale of these projects makes it difficult, tedious, and error-prone to determine by hand whether requirements have been fully met. This is especially true for large-scale high-assurance systems such as those found in aerospace and defense. For projects in these domains, millions of dollars in funding and even human lives may be at stake if project requirements are not correctly satisfied.

Use of automated techniques to reduce analyst effort and decrease the time it takes to complete the analysis *in conjunction with human activity* is warranted in such situations. In prior work (Hayes and Dekhtyar 2005), we discuss how a requirements tracing process can be improved for Independent Verification and Validation (IV&V) of mission-critical systems: human analysts submit a tracing task to an automated tool, obtain *a candidate RTM* from the tool, and then work (interactively or manually) on correcting the candidate RTM by *removing errors of commission* and detecting and fixing *errors of omission*. We view the automation of the satisfaction assessment process in a similar way. A human analyst submits a pair of artifacts and a certified (or a candidate) RTM to an automated satisfaction assessment tool. The tool produces a *candidate satisfaction assessment*. The analyst, either manually or working interactively with the tool, examines and possibly corrects the assessment (where necessary) and eventually certifies the candidate satisfaction assessment. As such, in approaching the satisfaction assessment problem, we use a roadmap similar to that established previously in (Hayes and Dekhtyar 2005; Hayes et al. 2006a, b) for the studies of automated traceability. The work to be conducted falls into two categories: the study of methods and the study of the analyst. In the first category (see (Hayes et al. 2003, 2006a, b) for the work of our group in the area of traceability), we study the automated means of producing good *candidate satisfaction assessments* given a pair of artifacts and an established RTM. The second part of the research (see Hayes and Dekhtyar 2005; Cuddeback et al. 2010) will address the work of the human analysts with candidate satisfaction assessments obtained from automated methods, and will study whether analysts can spot errors in the candidate assessments, and *what makes it easier for them* to do so. Our aim in this research is to assist analysts in performing satisfaction assessment. We posit that to do so, we must aim to produce accurate results, thus reducing the search space. For requirements deemed satisfied by the tool, the analysts will have to examine the returned matches to "certify" that the requirements are satisfied, but will not need to search through the entire set of design elements to do so.

In this work, we concretize the satisfaction assessment as the process of determining which pieces of a requirement are satisfied by which pieces of a set of design elements. In addition to providing a measure of system quality, determining the satisfaction assessment of a set of requirements and design elements provides information that allows one to see how potential maintenance efforts will affect a system, how well a system is suited to reuse on a future project, and how well an existing system meets both pre-existing and newly-introduced requirements.

In this paper, we study four methods for performing satisfaction assessment in an automated fashion: Naïve (Holbrook 2009; Holbrook et al. 2009), TF-IDF (Holbrook 2009; Holbrook et al. 2009), Rule-based (Holbrook 2009), and Combination (Holbrook 2009). We apply the methods to two datasets, measuring the recall, precision, number of corrections, selectivity, analyst verification effort, and f-measure in order to evaluate the effectiveness and efficiency of the methods.

We found that the TF-IDF satisfaction assessment method outperformed the naïve satisfaction assessment method (precision, selectivity, number of corrections). The rule-based satisfaction assessment method shows promise in discovering candidate satisfaction

mappings that are not found by other methods. The combination method, which uses both TF-IDF and rule-based approaches, has a higher level of recall than either method alone and reduces analyst effort.

This paper extends our previous work in (Holbrook et al. 2009) in a number of ways:

- A method to specify natural language rules is introduced.
- A combination method using information retrieval and natural language rule specification is introduced.
- An empirical study is presented that compares four automated satisfaction assessment methods using two data sets.
- A study of overall requirement satisfaction by design elements is presented, validated manually by two experienced software engineers and one NASA senior scientist.

Section 2 defines satisfaction assessment and presents the four methods, two of which were previously presented (Holbrook et al. 2009). Section 3 describes the study (includes two datasets) that was undertaken and previously reported (Holbrook et al. 2009) as well as presents additional results for the two new methods. Section 4 presents analysis and discussion. Section 5 discusses related work. Finally, Section 6 provides conclusions and future work.

2 Satisfaction Assessment

Satisfaction assessment is defined as the process of determining the satisfaction mapping of natural language textual requirements to natural language design elements. Given a set of requirements with each requirement, R, divided into unique terms *(R=(t_{r1}, t_{r2}, ...))* or phrases *(R=(p_{r1}, p_{r2}, ...))* and a set of design elements with each design element, D, divided into unique terms *(D=(t_{d1}, t_d2, ...))* or phrases *(D=(p_{d1}, p_{d2}, ...))*, a *satisfaction mapping* is a set of pairs of terms (t_{rn}, t_{dm}) where t_{rn} is a term in a set of requirements and t_{dm} is a term in the set of design elements where tr_n is directly correlated to t_{dm}. A satisfaction mapping may also occur at the phrase or chunk level, consisting of a series of chunk pairs, (c_{rn}, c_{dm}) with c_{rn} being a phrase in a requirement and c_{dm} being a phrase in a corresponding design element (where c_{dm} directly addresses c_{rn}). Here, we have broken down the requirement and design element text into phrases based on parts of speech. We refer to each of these phrases as requirement chunks or design element chunks. Each chunk has its own unique identifier. We refer to satisfaction assessment at the phrase level as chunk-level satisfaction assessment. In this study, we look at satisfaction assessment at the chunk level and also at the requirement level. Requirement-level satisfaction involves determining whether individual requirements are satisfied by corresponding design elements. In order to make this determination, we analyze chunk level satisfaction mappings and apply requirement labels – satisfied or not satisfied – in a binary classification technique.

Satisfaction assessment, as performed in this study, consists of a variety of processing techniques. Requirements and design elements are accepted in their natural form. No formatting, language rules, or models are imposed, so as to not burden the analysts. Also, the method does not attempt to examine possible underlying abstract models. The sole input requirement is that documents be in English text. An RTM for each dataset was also used as input in this work. After breaking requirements and design elements into chunks, the search space becomes the Cartesian product of the number of requirement and design element chunks. To reduce this very large search space, the RTM is used to only compare requirement chunks to design element chunks that are at least related by the RTM through tracing.

2.1 Satisfaction Assessment Process

Satisfaction assessment is a three step process. The first step is to parse the requirement and design elements into chunks. This step is described in Section 2.2.

The second step is to map design chunks to the requirement chunks to which they are similar, resulting in a candidate satisfaction assessment mapping. This is viewed as a mapping problem, which can be defined as an information retrieval (IR) problem: given a document collection and a query, determine those documents from the collection that are relevant to the query. Our prior work, and that of others, has shown that requirement and design similarity can be modeled, or at least approximated, by the document relevance notions on which different IR algorithms rely (Cleland-Huang et al. 2005; Greenspan et al. 1994; Hayes and Dekhtyar 2005; Hayes et al. 2003; Cleland-Huang 2002; Spanoudakis et al. 2004). Methods applied to the mapping step are described next.

In the third step of the satisfaction assessment process, chunk-level satisfaction assessments are processed in an automated fashion to determine satisfaction at the requirement level. The output of this requirement-level satisfaction process is a series of requirements, each with labels applied to indicate whether the requirements are satisfied or are not satisfied. The results from each level are shown to the analyst for approval. In this study, we compared chunk-level candidate satisfaction assessments to a golden answer set, created by analysts (the answer set creation process is described below in Section 3.1). Additionally, we gathered analyst input on the final requirement-level satisfaction assessments.

2.2 Processing

Prior to chunk-level satisfaction assessment, a series of processing steps are applied to the requirements and design elements before a candidate satisfaction assessment can be determined. First, text is divided into chunks. This step was performed using OpenNLP tools (OpenNLP). Chunking takes place by parsing sentences, with each phrase of the sentence identified uniquely as a chunk. Incremental processing of text is shown in Fig. 1. The chunking process takes on the order of seconds for small data sets and less than a minute for the larger CM-1 data set in this study.

Next, additional preprocessing is performed (stop word removal and stemming). A domain-specific thesaurus is developed and applied, containing a set of synonym pairs for domain-specific vocabulary (the thesaurus was developed by analyzing a subset of 25% of the requirements and design elements for each dataset domain). Any acronyms encountered are entered as full text in the thesaurus. Thesaurus entries are of the form "term synonym1 synonym2… synonymN" where term is synonymous with synonyms 1 through N. Note that terms can be contained in two distinct synonym groups. For example, if the term "space" is used in the thesaurus as both an area and an expanse in which planets reside, the analyst could create distinct thesaurus entries such as:

```
space area place

space heavens universe
```

During thesaurus tagging, each synonym is replaced with a unique term that represents a particular synonym set. Next, each requirement and design element is tokenized into chunks based on parts of speech.

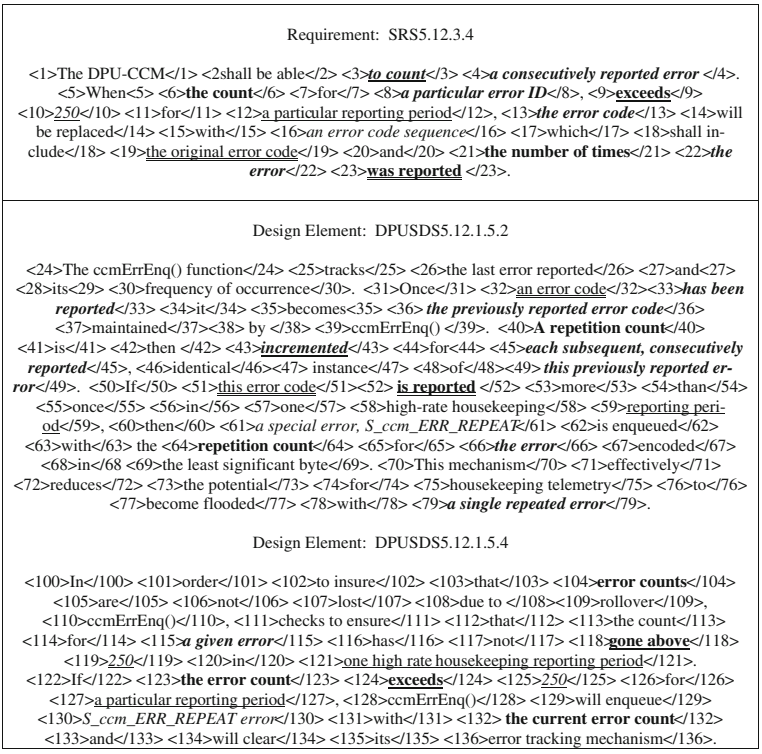


Fig. 1 Sample requirement and design element satisfaction assessment

A sample satisfaction mapping is shown in Fig. 2. The text in Figs. 1 and 2 has been formatted to visually show the matches (e.g., Fig. 2 indicates that requirement chunk 19, "the original error code," maps to design chunk 32 in design element 1, "an error code."). To make it easier to see related items, each is double-underlined. Similar formatting changes (bold, italic and underline combinations) indicate other potential mappings between requirement element chunks and design element chunks. Finally, similarity measures are calculated between chunks of requirements and the design elements to which they are tied in the project RTM. The similarity measures for this study are determined using one of the methods described next. 2.2.1 Naïve satisfaction assessment. The naïve satisfaction approach examines textual similarity and is our baseline method. If terms within a requirement chunk and design chunk in the dataset contain the same root or the root of a synonym, then the terms are considered a match. The overall percentage of matching terms in a chunk, excluding stop words, is the weighted similarity value for a requirement and design element chunk. For example, for two chunks c_{rn} and c_{dm}, where each consists of a series of terms, the similarity score for naïve satisfaction is:

$$\text{sim} = \frac{(\text{number of common terms in } (c_{rn} \text{ and } c_{dm})}{(\text{number of terms in } c_{rn} + \text{number of terms in } c_{dm})}$$

Threshold values from 0.01 to 0.9 (described in Section 2.2.1) are used to filter chunks, chunks with similarity values below the threshold do not appear in the candidate satisfaction assessment mapping. The algorithm chunk-level for naïve satisfaction is shown in Fig. 3.

Fig. 2 Sample requirement and design element satisfaction assessment (cont'd)

Satisfaction Assessment:
1
2
3 - 43 (***bold underline italic***)
4 - 33, 36, 45, 49, 66, 79, 115 (***bold italic***)
5
6 - 40, 64, 104, 123, 132 (**bold**)
7
8 - 33, 36, 45, 49, 66, 79, 115 (***bold italic***)
9 - 118, 124 (**bold underline**)
10 - 119, 125 (*underline italic*)
11 - 126
12 - 59, 121, 127 (underline)
13 - 33, 36, 45, 49, 66, 79, 115 (***bold italic***)
14
15
16 - 61, 130 (*italic*)
17
18
19 - 32, 51 (double underline)
20
21 - 40, 64, 104, 123, 132 (**bold**)
22 - 33, 36, 45, 49, 66, 79, 115 (***bold italic***)
23 - 52 (**bold double underline**)

2.2.1 Satisfaction Assessment Using TF-IDF

TF-IDF is a statistical measurement of the importance of a term within a document. Term frequency (TF) is the (possibly normalized) number of times a term (or word) appears within a document. Inverse document frequency (IDF) of a term is the logarithm of the ratio of the total number of documents in a collection to the number of documents that contain the term. The less frequent a term is, the more discriminating power it has, and thus the higher the IDF [21]. The weight of each word in a document is the product of TF and IDF (Baeza-Yates and

```
Naïve satisfaction( )
        Load an RTM for the data set
        Load requirement and design element text
        Set a threshold for naïve satisfaction method, thresh

        For each requirement and each design element
                Perform stopword removal
                Perform stemming
                Chunk text into phrases
                Tokenize text into individual terms
                Tag each term with synonyms

        For each requirement i
                For each chunk m in requirement i
                        For each design element j that is mapped to requirement i in the RTM
                                For each chunk n in design element j
                                        Calculate number of terms, numMatches, that chunk m has in common
                                        with chunk n
                                Calculate total number of terms in chunk m and all chunks n, totalNumTerms
                                PercentageMatch = numMatches/totalNumTerms

                                if(PercentageMatch > thresh)
                                        Mark chunk m and chunk n as a candidate satisfaction mapping pair,
                                        with confidence PercentageMatch
```

Fig. 3 Naive satisfaction assessment algorithm

Ribeiro-Neto 2003). Formally, each document *d* is represented as a vector $v_d=(w_1,\ldots,w_m)$ of term weights. Term weights are computed as follows:

$$w_i = \left(n_i / \max\left(n_j\right)\right) \cdot \log(|D|/|\{d|d\ t_i\}|)$$

where n_i is the number of occurrences of term t_i in the document and, $max(n_j)$ is the maximum frequency of a single term in the current document, |D| is the total number of documents, and $|\{d:d \ni t_i\}|$ is the number of documents containing term t_i. Vectors containing these weights are constructed (Baeza-Yates and Ribeiro-Neto 2003).

Two vectors, v_r and v_{rd} , representing two documents are compared using the *cosine similarity measure*, which is, essentially the cosine of an angle between the two vectors:

$$sim(v_d, v_r) = \frac{v_r \circ v_d}{||v_r|| \cdot ||v_d||}$$

Here, the numerator is the dot-product of the two vectors and the denominator is the product of the lengths of the vectors.

We use vectors at the chunk level for this work. Similarity scores are between 0 and 1, where 0 indicates that the two documents have no relevancy and 1 indicates that the documents are identical (Baeza-Yates and Ribeiro-Neto 2003).

Each requirement and design element chunk is considered an individual document within the document collection. TF-IDF similarity scores are calculated between pairs of chunks (c_{rn}, c_{dm}), where c_{rn} is the n^{th} requirement chunk in requirement R and $c_{dm}$is the m^{th} design element chunk in design element D, and where requirement R is mapped to design element D in the RTM for the dataset. All such pairs from the RTM are considered to potentially be matches. If the pair (c_{Rn}, c_{Dm}) has a similarity score above a given threshold value (we used 18 threshold or filter values – nine values starting at 0.01 and incrementing by 0.01 to 0.09, and nine values starting at 0.1 and incrementing by 0.1 to 0.9), then the two are considered a satisfaction match and the pair (c_{Rn}, c_{Dm}) is included in the candidate satisfaction assessment mapping. The entire set of satisfaction match pairs that have similarity scores greater than the threshold value is considered to be a candidate satisfaction assessment for a given dataset. We call our final output a candidate mapping because we acknowledge that an analyst should examine the tool's output and confirm the final mapping (Hayes and Dekhtyar 2005). The algorithm for TF-IDF chunk-level satisfaction assessment is shown in Fig. 4.

```
TF-IDF satisfaction( )
    Load an RTM for the data set
    Load requirement and design element text
    Set a threshold for TF-IDF satisfaction method, thresh

    For each requirement and each design element
        Perform stopword removal
        Perform stemming
        Chunk text into phrases
        Tokenize text into individual terms
        Tag each term with synonyms

    For each requirement i
        For each chunk m in requirement i
            For each design element j that is mapped to requirement i in the RTM
                For each chunk n in design element j
                    Calculate TF-IDF similarity measure between chunk m and chunk n, sim
                    if(sim > thresh)
                        Mark chunk m and chunk n as a candidate satisfaction mapping
                        pair, with confidence sim
```

Fig. 4 TF-IDF satisfaction assessment algorithm

2.2.2 Natural Language Processing Rule-Based Satisfaction Assessment

After the text has been processed and chunked, the parts of speech of each term within a textual requirement or design element are determined through parts of speech tagging algorithms. Parsing reveals information about sentence structure. After investigation of a number of potential NLP toolkits, we decided to use OpenNLP (OpenNLP). The API was used to parse sentences into chunks based on parts of speech (i.e., noun phrase, verb phrase, etc.) and to perform tagging of the part of speech (see below).

From this collection of information about the text, we can determine a satisfaction assessment between a set of requirements and design elements. Rules may be created that help identify requirement-design element chunk pairs that should be included in a candidate satisfaction assessment. For example, consider the requirement, "the system shall allow incremental processing of information from the DPU-CCM," and the design element "First, the DPU-CCM will transmit information to the TMALI CSC. Next, that information will be broken down into individual packets and transmitted to subsystems accordingly." The word "incremental" from the requirement may not map directly to the terms "first" and "next" within the design element, but a human analyst may notice this relationship. Through the application of natural language processing algorithms, one will see that "incremental" is an adjective describing processing.

The terms "first" and "next" act as transitional phrases within the sentence. One example of an NLP rule that could be applied is: "If a requirement subject is described as incremental, piece-wise, step-by-step, or any synonym of these and subsequent design element(s) describe steps or contain a series of transitional phrases, then the incremental nature of the requirement is captured in the subsequent design element(s)." This would then return matches between "incremental" and "first," and between "incremental" and "next." The combination of NLP tagging and thesaurus entries tying terms such as "incremental" and "first" allow these types of relationships to be captured.

Rules A rule specification interface was created within our tool (see Section 2.2.4) to create and maintain rule sets. Analysts may also create rule sets manually using a text editor. Rule sets may be processed individually or in batch mode. In batch rule processing mode, a file containing a list of rule sets is loaded. Each rule set is processed individually for each of the threshold values specified.

Rules can be based on any permutation of grammatical tagging and placement. A rule is specified in the following format:

[Element1Position]|[Element1PartofSpeech]|[Element1Type]|[Element2Position]|[Element2PartofSpeech]|[Element2Type]|
[MinSimilarity]|[Confidence]|[Enabled].

For example, the rule:

`Any|NP|RE|First|VP|DE|45|20|True`

corresponds to an active rule that specifies that if any noun phrase in a requirement chunk is at least a 45% match based on lexical similarity with the first verb phrase in a design element, then the requirement chunk and design element chunk should be paired with 20% confidence. Possible values for each of the elements listed above are shown in Table 2.

The parts-of-speech tagging provided by OpenNLP does not have a one-to-one correspondence to the tagging system used by our tool (OpenNLP). OpenNLP uses the University

of Pennsylvania Penn Treebank Tagging system (Marcus et al. 1993). Table 1 indicates the tagging correlations made for this research and the corresponding requirements tool created, Requirements SATisfaction (RESAT). The algorithm for NLP rule-based chunk-level satisfaction assessment is shown in Fig. 5.

During testing phases of this research, 358 sets of rules at 18 threshold values were considered. Rules were determined by manually inspecting a small subset of one of the datasets and by analyzing textual patterns in requirements and design elements that were not used as input data in this research.

Possible Rules and Rule Sets Extensive analysis beyond the results presented in this paper has been performed on the rule-based approach for satisfaction assessment. These represent a subset of all possible rules that can be specified with our tool.

The Rule Space A rule can be created with 6° of freedom (element 1 position, element 2 position, element 1 part of speech, element 2 part of speech, minimum similarity, and confidence). Table 2 shows the possible input values for each of these degrees. To calculate the size of the rule space, the following assumptions were made:

- A requirement or design chunk contains at most 10 tokens that have the same part of speech,
- Adequate similarity coverage can be obtained if one looks at each percentile of similarity (1% similarity, 2% similarity… 100% similarity),
- A minimum similarity percentage of 0 is meaningless in assisting analysts with satisfaction assessment because all possible matches will be returned, and
- Adequate confidence coverage can be obtained if one looks at each percentile of similarity (1% similarity, 2% similarity… 100% similarity) at the actual weight as the confidence value, and at "2,3,4…100×" the actual similarity (weighting confidence as n times the actual similarity is useful for weighting some rules more heavily).

Each of these rule sets can be tested against a threshold value ranging from 0 to n. The threshold is used to determine the total minimum weight required to include a requirement-design element chunk pair in the final candidate satisfaction assessment and is compared against the sum of all weights returned for all rules run on that requirement-design element chunk pair.

Table 1 RESAT and Penn Treebank parts of speech tags

Part of Speech (RESAT Tag)	Penn Treebank Tags
Noun Phrase (NP)	NN, NNP, NNPS, NNS, PRP, WP
Verb Phrase (VP)	VB, VBD, VBG, VBN, VBP, VBZ
Adjective Phrase (ADJP)	JJ, JJR, JJS
Adverb Phrase (ADVP)	RB, RBR, RBS
Conjunction (CC)	CC
Interjection (INTJ)	UH
List Marker (LST)	LS
Prepositional Phrase (PP)	IN
Particle (PRT)	RP
Word (WORD)	All

```
NLP Satisfaction( )
Load an RTM for the data set
Load requirement and design element text
For each requirement and each design element
        Perform stopword removal
        Perform stemming
        Chunk text into phrases
        Tokenize text into individual terms
        Tag each term with synonyms
        Tag each term and chunk with parts of speech information

Load or specify user-defined and standard rules for satisfaction assessment into a rules set, rules

For each requirement i
        For each chunk m in requirement i
                For each design element j that is mapped to requirement i in the RTM
                        For each chunk n in design element j
                                For each rule r in rules
                                        if(r holds true for chunks m and n)
                                                Mark chunk m and chunk n as a candidate sat-
                                                isfaction mapping pair, with confidence provided
                                                by rules assessment.  If multiple rules hold, av-
                                                erage the weights provided by each rule.
For each requirement i
        For each chunk m in requirement i
                For each design element j that is mapped to requirement i in the RTM
                        For each chunk n in design element j
                                If the weight for the requirement chunk-design element chunk pair is
                                greater than the threshold value specified, add this pair to the candidate
                                satisfaction assessment
```

Fig. 5 NLP satisfaction assessment algorithm

Rule Families Results presented in this paper are limited in scope to rule sets containing a single rule. Multiple-rule sets and other test runs are documented in Holbrook's dissertation[1] (Holbrook 2009). These rules were divided into four rule families, based on the parts of speech tested. The four rule families are described in Table 3.

2.2.3 Combination Satisfaction Assessment: Natural Language Processing Rule-Based and TF-IDF Approach

Upon analysis of the results of the rule-based method, it was clear that while no single rule set analyzed achieved the recall of the TF-IDF approach, the rule-based approach returned several correct answer set mappings that were not found through the TF-IDF approach. Thus, a fourth method was developed that combined the TF-IDF and rule-based approaches.

The Combination Approach first determines the satisfaction assessment of a dataset using TF-IDF. Next, the satisfaction assessment of the dataset is determined using the Rule-based approach. Finally, the answer sets are combined to create a final answer set. During the

Table 2 Possible rule values

Factor	Range of Values
Element 1 Position	{ Any, First, Second, … N^{th}}
Element 2 Position	{ Any, First, Second, … N^{th}}
Element 1 Part of Speech	{ NP, VP, ADJP, ADVP, CONJP, INTJ, LST, PP, PRT, WORD }
Element 2 Part of Speech	{ NP, VP, ADJP, ADVP, CONJP, INTJ, LST, PP, PRT, WORD }
Minimum Similarity	0.0–100.0
Confidence	−100.0 to 100.0, including 0

Table 3 Rule families

Rule Family	Description	Rules
RF1	Rules based on core parts of speech (noun phrases and verb phrases)	d_nounphrase r_nounphrase d_verbphrase r_verbphrase
RF2	Rules based on descriptive parts of speech (adjective phrases and adverb phrases)	d_adjectivephrase d_adverbphrase r_adjectivephrase r_adverbphrase
RF3	Rules based on combinatorial parts of speech (conjunctions and list markers)	d_conjunction d_list r_conjunction r_list
RF4	Rules based on auxiliary parts of speech (interjections, particles, and prepositions)	d_interjection d_particle d_preposition r_interjection r_particle

combination process, weights returned by the two methods were averaged and duplicate candidate satisfaction mappings were eliminated. See Section 3.1 for more information on answer set construction and data analysis. The algorithm for the Combination Approach at the chunk level is shown in Fig. 6.

To facilitate the specification of NLP rules by a human analyst (and to implement the TF-IDF, naïve, and combined methods), a tool was developed. The tool is described next.

2.2.4 RESAT Tool

The software implementation of the satisfaction assessment procedures is called REquirements SATisfaction (RESAT). From within the RESAT tool, users can load a set of requirements and a set of design documents, a domain thesaurus, set threshold values, and perform automated satisfaction assessment. Batch mode processing is also available within the tool to process datasets at multiple threshold values. The rule specification interface is shown in Fig. 7.

RESAT allows analysts to generate initial candidate satisfaction assessments given a set of inputs. This can then be used to highlight areas of potential ambiguity or weak satisfaction assessment, as well as to indicate clear requirement satisfaction assessment relationships. Figures 8, 9 and 10 depicts the RESAT visualization of candidate satisfaction assessments. This will be used by the analyst to perform vetting. We anticipate that for data sets with high densities of domain specific terms, the analyst will also want to create a domain specific thesaurus as input. This portion of work is reusable on future satisfaction assessment within the same or a similar domain, but is not helpful when applied to a domain with varying terminology. For example, many terms and acronyms have varying meaning between different domains, and thus have different synonyms.

```
NLP+TF-IDF satisfaction( )
    Load an RTM for the data set
    Load requirement and design element text
    Set a threshold for the TF-IDF method

    For each requirement and each design element
        Perform stopword removal
        Perform stemming
        Chunk text into phrases
        Tokenize text into individual terms
        Tag each term with synonyms
        Tag each term and chunk with parts of speech information

    Load or specify user-defined and standard rules for satisfaction assessment into a rules set, rules

    For each requirement i
        For each chunk m in requirement i
            For each design element j that is mapped to requirement i in the RTM
                For each chunk n in design element j
                    For each rule r in rules
                        if(r holds true for chunks m and n)
                            Mark chunk m and chunk n as an NLP candi-
                            date satisfaction mapping pair, with confidence
                            provided by rules assessment. If multiple rules
                            hold, average the weights provided by each
                            rule.
    For each requirement i
        For each chunk m in requirement i
            For each design element j that is mapped to requirement i in the RTM
                For each chunk n in design element j
                    Calculate TF-IDF similarity measure between chunk m and chunk n, sim
                    if(sim > thresh)
                        Mark chunk m and chunk n as a TF-IDF candidate satisfaction
                        mapping pair, with confidence sim
    For each requirement i
        For each chunk m in requirement i
            For each design element j that is mapped to requirement i in the RTM
                For each chunk n in design element j
                    If the weight for the requirement chunk-design element chunk pair is
                    greater than the threshold value specified in either the NLP candidate
                    answer set or the TF-IDF candidate answer set, add this pair to the can-
                    didate satisfaction assessment
```

Fig. 6 NLP+TF-IDF satisfaction assessment algorithm

3 Evaluation of Satisfaction Assessment Methods

We undertook a study using two datasets. The study design, measures, and threats to validity are presented below.

3.1 Study Design

Two industry projects were used for the study. Each dataset consisted of textual requirements (high-level elements) and textual design documents (low-level elements). NASA CM-1 (Predictor Models in Software Engineering) consists of the entire requirement specification (235 requirements) and the entire design document (220 design elements) for a NASA scientific instrument. After the dataset was chunked, there were a total of 2,780 requirement chunks and 10,490 design chunks. The requirements traceability matrix (RTM) for this dataset contains 362 links between requirements and design elements, with a density of 1.54 design elements per requirement. It should be noted that the RTM is sparse, meaning that not all requirements in the dataset have corresponding design elements. Using the RTM, there were 205,696 requirement-design element pairs to be analyzed. Without the RTM, considering every possible requirement-design element pair, there would have been 29,162,200 comparisons.

GanttProject, an open source program, creates Gantt charts and performs basic project management (GanttProject). There are 17 requirement elements and 78 design elements. After chunking, there were 312 requirement chunks and 632 design chunks. The RTM for

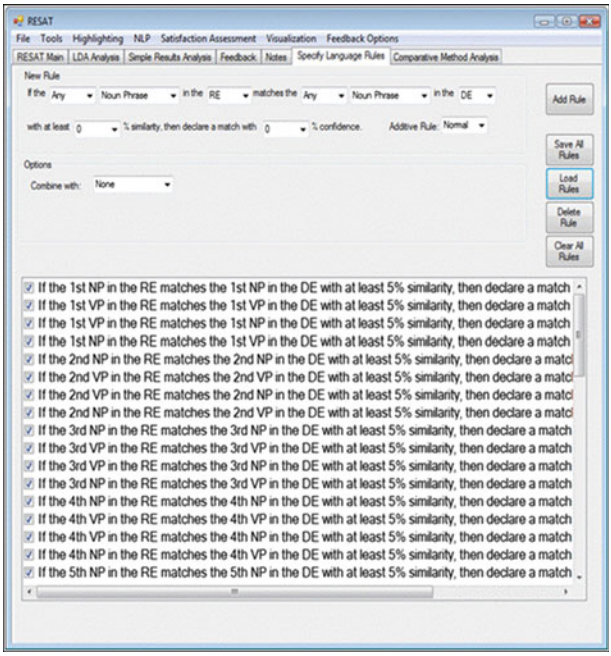


Fig. 7 RESAT rule specification interface

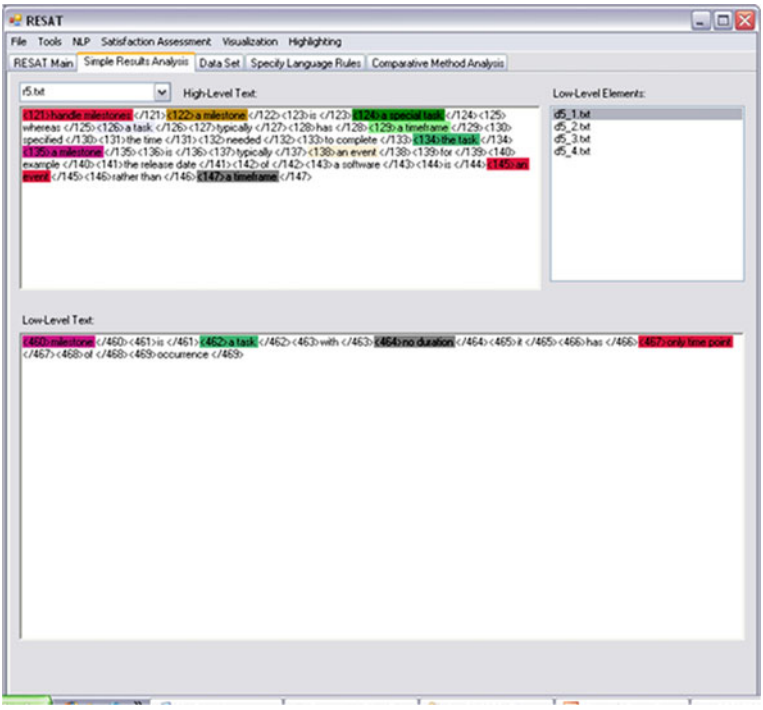


Fig. 8 RESAT candidate satisfaction assessments visualization interface

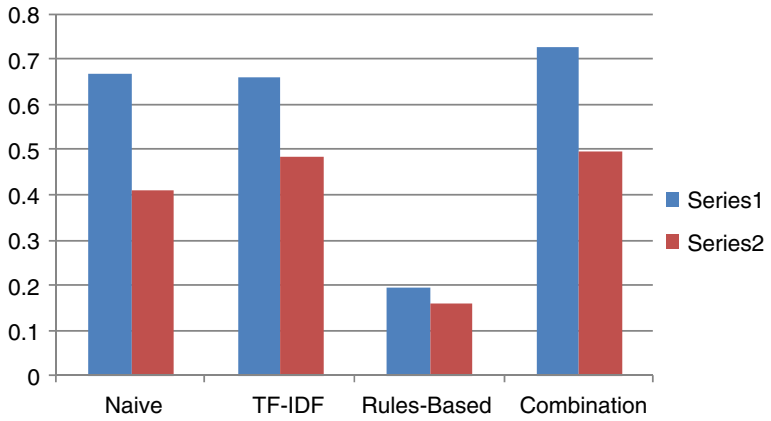


Fig. 9 Recall and precision for all methods with the Gantt data set

the GanttProject dataset contains 68 links, with a density of 4.0 design elements per requirement. From these, there were 15,430 requirement-design element pairs to be analyzed. Without the RTM, considering every possible requirement-design element pair, there would have been 275,064 comparisons to be made. The authors previously introduced a taxonomy of project sizes for the purpose of performing traceability tasks (Hayes et al. 2006a, b). According to this taxonomy, the GanttProject dataset is a medium project, while the CM-1 dataset is a large project (Hayes et al. 2006a, b). In our experience, the CM-1 dataset is rather typical of software artifacts generated for various NASA instruments. Example requirement elements and design elements from both the CM-1 and Gantt datasets can be found in Appendix A.

In order to validate the accuracy of our methods, golden answer sets were built for the datasets at the chunk level. Two analysts (not the authors) built the answer sets from the chunked text for each dataset. The two analysts were experienced graduate students. One had roughly 7 years of industry experience, the other had academic experience only at the time. Both had 4 years of experience in performing software engineering research. One analyst constructed the initial answer sets, while the second analyst reviewed and offered suggestions as

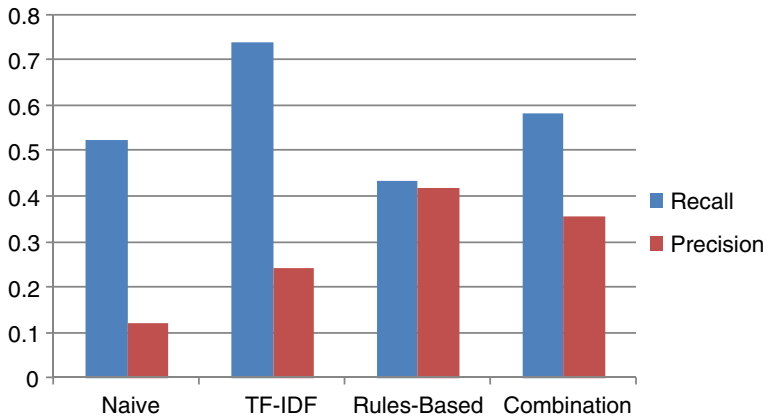


Fig. 10 Recall and precision for all methods with the CM-1 data set

necessary. As only one analyst built the answer set, inter-rater reliability statistics could not be applied. The analysts met and reviewed the suggestions to produce final answer sets. The golden satisfaction answer set for CM-1 has a total of 959 satisfaction mappings, with an average density of 0.09 design element chunks per requirement chunk. The golden satisfaction answer set for GanttProject has a total of 307 satisfaction mappings with a density of 0.98 design element chunks per requirement chunk. It took the analysts a combined total of 15 h to create the initial Gantt answer set, and another 4 h for verification. Analyst one spent 120 h creating the initial CM-1 answer set, and analyst two spent another 40 h on verification. For the requirement-level satisfaction study, we validated our answer sets with the help of a NASA expert with 35 years of NASA experience. The NASA expert created the requirement-level answer set for Gantt. The NASA expert also reviewed a random sampling of 130 of the requirements from CM-1. These were reviewed at the requirement level and the analyst was asked to ensure that the requirements were satisfied by the design. For CM-1 items that were reviewed by the NASA expert, his view of inclusion of values in the answer set took precedence over the student analysts' answers. For CM-1 items that were not evaluated by the NASA expert, we only added items to the answer set if the opinions of the remaining participants were unanimous (both agreed that the requirement was satisfied or both agreed that the requirement was not satisfied). Empirically, it took the analysts roughly 1/2 to 2 min per requirement to verify results, whereas the creation of an answer set without the tool was a much more arduous and time consuming task.

Each method from Section 2.2 was applied to each dataset, using the measures discussed in Section 3.2, in order to produce candidate chunk-level satisfaction assessments. In addition, the candidate chunk-level satisfaction assessments were run through a post-processor to output a satisfaction requirement-level candidate satisfaction label ('satisfied' or 'not satisfied') for each requirement. These candidate answer sets (for the chunk level and requirement level studies) were then compared to the golden answer sets described above.

3.2 Measures

Our goal is to assess the precision of the methods in determining the mappings between the requirement and design element chunks and to examine overall accuracy at the requirement level. Three traditional information retrieval measures are used: precision, recall, and f-measure. Given a list of candidate satisfaction pairs, the precision of the list is the percentage of the retrieved pairs that are correct. Recall of the list is the percentage of correct pairs retrieved.

High precision means low incidence of type I errors (including incorrect pairs). Higher precision indicates that analysts will have fewer incorrect results to remove in order to obtain a true satisfaction assessment. High recall means low incidence of type II errors (omitting true pairs). High recall indicates that a majority of the true matches were returned, meaning reduced work for the analyst (will not have to spend as much time searching for satisfaction mappings that have been omitted from a candidate satisfaction assessment).

Given a list of candidate satisfaction pairs, the pair (precision, recall) provides a good description of the accuracy of the list. However, when the accuracy of different lists needs to be compared, it is much more convenient to combine precision and recall into a single measure. This is typically done via f-measure (Baeza-Yates and Ribeiro-Neto 2003), the harmonic mean of recall and precision. We use a variant of f-measure called f_2 (or f2), computed as follows:

$$f2 = \frac{3 \cdot precision \cdot recall}{(2 \cdot precision) + recall}.$$

The f2 measure is a weighted harmonic mean of recall and precision which favors recall. We favor recall in our evaluation because traditionally the cost of repairing type II errors is higher than the cost of repairing type I errors (Hayes and Dekhtyar 2005). Thus, when comparing two lists of candidate satisfaction pairs with the same number of total errors, we give preference to the list with fewer type II errors, i.e., with higher recall. At the requirement level, we examine precision and accuracy and produce confusion matrices (included in Appendix B), for each method and data set combination.

3.3 Threats to Validity

There are external threats to validity that may impact the generalizability of our results. Our methods were applied to only two systems and two domains. Further research on additional data sets could be used to determine the applicability of this research across domains. Experimentation with varying data set sizes could help us understand which rule sets and methods perform best on small, medium and large data sets. Performing studies on alternate domains could help understand the extent to which this research can be generalized. Additionally, research into whether rule sets can be tailored for one domain/data set size and then applied to an alternate domain/data set size would be interesting to determine the extent to which these results can be generalized. One way to increase external validity is to enable other researchers to replicate the study. Toward that end, we mitigated reliability threats to validity. Our process is outlined. Our datasets are available upon request and we plan to make the RESAT tool available upon request, pending University approval. We believe the study is repeatable.

We attempted to minimize threats to internal validity. The datasets chosen were not reorganized or modified, the preprocessing steps were constant between methods, and the same methods were applied to both datasets. Thesaurus entries were created based on the same percentage (25%) of elements examined. The analyst who created the thesaurus was familiar with requirements engineering, but not well-versed in the specific domains for CM1 and Gantt.

With respect to construct validity, it is possible that our design suffered from mono-method bias. The analysts that built the answer sets for the chunk-level study were not subject matter experts on the systems, and we did not have multiple groups of experts build answer sets and then compare them. It is possible that a different group of analysts may yield different golden answer sets. We attempted to mitigate this threat via the answer set creation process discussed in Section 3.1. During golden answer set creation for the Gantt data set, two analysts collaborated. Analyst 1 created the initial satisfaction answer set for ten of the requirements, and Analyst 2 created the initial satisfaction answer set for the remaining seven requirements. After creating the initial satisfaction answer set, the analysts traded results and verified the other's work. With any difference in opinion, the analysts discussed and came to consensus. For the CM-1 data set, Analyst 1 created the initial satisfaction answer set for the entire data set and Analyst 2 verified the initial answer set. Again, where there was disagreement, the analysts discussed and came to consensus. For both data sets, a third individual verified the process, but did not verify the final answer set. For the requirement-level study, we used the NASA expert to build answer sets, conservatively augmenting his results only with unanimous opinions of the other participants. The NASA expert was quite familiar with the domains of both datasets. The remaining participants were very familiar with the Gantt dataset domain and were familiar with the CM-1 data set domain. Still, it is possible that different experts or participants could generate different answer sets.

We attempt to mitigate other construct validity threats by using real world requirements and design elements that were not specifically created for this work. In doing this, the quality

of the input data set is also a potential threat to validity in that requirements written in well-formed English will be more accurately classified by the NLP tools and thus may generate higher quality candidate satisfaction assessments. It is possible that the baseline TF-IDF method could have been particularly well- or ill-suited to our data sets. We chose this as our baseline IR method due to its widespread use in data mining and tracing research and its relative success within those applications. Finally, bias could have been introduced through the creation of NLP rules for the Rule-based and Combination methods. We attempted to mitigate this threat to validity by examining only a small subset of the data set before creating the rules. We did not modify or add rules as the study proceeded. In a future study, we will have an analyst unfamiliar with the dataset build all rules.

Bias may have been introduced through our merging of method results in building a single labeled list of requirements for the requirement-level validation. By including any requirements labeled as satisfied by any method, there could be a high number of false positives. However, we wanted to get opinions of experts on as many items as possible, so we used this broader criterion for selection for evaluation.

3.4 Satisfaction Evaluation

To assist analysts, a requirement-level candidate satisfaction label is provided for each requirement and its mapped design elements: *satisfied* or *not satisfied*. To do this, we performed post-processing of the candidate satisfaction mappings by looking at the number of chunks in the requirement (L) and the number of those chunks with corresponding matching design elements (M). These values were used to calculate a satisfaction rate for each requirement: M/L. For example, consider a requirement with 10 chunks, seven of which have been marked as satisfied by design elements in the candidate answer set. The satisfaction rate for that requirement would be 70%. After each requirement's satisfaction rate was calculated, a threshold was applied. Requirements above that threshold were considered satisfied, requirements below the threshold were considered not satisfied. For this study, we applied a 75% threshold to determine satisfaction at the requirement level.[2] This was done for the output of each method. The results were then consolidated to ease manual review (if any of the methods labeled a requirement satisfied, it was so labeled). We refer to the output of the post-processing as requirement-level candidate satisfaction labels.

To validate our requirement-level candidate satisfaction results for each method, we enlisted three outside parties to perform evaluation. One validation participant is a NASA senior scientist with 35 years of NASA experience. The other two validation participants each have at least 11 years of software engineering experience in industry. Each validation participant was asked to independently review every requirement deemed satisfied by the tool for each dataset. In addition, they were asked to review roughly 60% of the requirements deemed not satisfied by the tool.[3] In addition to hardcopy printouts of the text, participants had access to RETRO.NET (Hayes et al. 2007). They were permitted to use it as a search agent to display the requirements and design elements and/or to search within the full set of design elements in order to find missing satisfaction assessment matches.

[2] This threshold was determined based on discussions with IV&V analysts (not involved in the assessment study). Also, when other thresholds were used, it was the case that the number of requirements labeled as satisfied was unrealistically small or large (no requirements satisfied, all requirements satisfied).

[3] We did not ask the participants to review every 'not satisfied" requirement due to participant time constraints. The participants were assigned review items so as to achieve coverage while also ensuring as much overlap as possible (for example, having reviewer 1 look at the first 30 items and reviewer 2 look at items 20 – 40).

3.5 Results

The results for candidate satisfaction mappings at the chunk-level are presented first followed by evaluation of the requirement-level satisfaction labels.

3.5.1 Results for Chunk-Level Candidate Satisfaction Mappings

All four rule-based methods were applied to the chunked requirements and design elements of each dataset. The lists of candidate satisfaction pairs were constructed for threshold parameters with values 0.01, 0.02 through 0.09, and from 0.1, 0.2 through 0.9. Threshold values were chosen based on prior observation that high f2 measures occur within these ranges. Precision, recall, and f2 were computed for each threshold value. The complete results of our chunk-level study are shown in Tables 7, 8, 9, 10, 11, 12, 13 and 14 in Appendix B at the end of the paper. Each table documents recall, precision, and f2 for each threshold value considered by the method. The key results of the study are summarized in Tables 4 and 5. For each dataset and method, we report the largest value of the threshold (where appropriate) at which the best value of the f2 measure was achieved, listing also the precision, recall, and f2 values for that threshold. For example, we can see that the largest threshold value with the best value of f2 occurred at threshold 0.2 for TF-IDF for the Gantt dataset with recall of 0.664, precision of 0.486, and f2 measure of 0.619. The methods that involve rules (rule-based and combination) do not list threshold value but list the rule family (1 through 4) and the actual rules that resulted in the highest f2 measure. For example, the best f2 value for the Combination method for Gantt dataset was rule d_adverbphrase in Rule Family 2 with recall of 0.73, precision of 0.499, and f2 of 0.632. In each of the other three rule-based cases, more than one rule achieved the same f2 value (for example, the rule-based method for the CM-1 dataset had three rules from rule family 1 achieve an f2 measure of 0.430). Additionally, we include the cases when significantly higher recall (albeit at lower f2 value) was achieved.

Gantt Dataset We first examine the performance of the methods on the Gantt dataset (Tables 7, 8, 9 and 10 in Appendix B). The recall values are almost identical for TF-IDF and naïve, with a high of 0.674. Recall values for the rule-based method are very low, topping out at a mere 0.198. The combination method greatly boosts recall, with values ranging from 0.667 to 0.729. Even the lowest recall values for the combination method are acceptable. This is due to the TF-IDF method retrieving many true links.

Precision is somewhat higher for TF-IDF at 0.486 as compared to 0.41 for naïve. Precision is low for the rule-based method with a high of 0.31 (but with recall of 0.138), as judged by guidelines in prior work (Hayes et al. 2006a, b). The combination method increases precision in all cases, with a high of 0.569 (with recall of 0.667). The f2 measure for TF-IDF is better than for naïve at 0.619 compared to 0.539. Overall, the TF-IDF method outperforms the naïve method and the rule-based method for the Gantt dataset. Only the combination method outperforms TF-IDF.

That aside, it is worth noting that three of the four methods fared reasonably well. The recall values for each method, except rule-based, are acceptable (above 65%) and the precision values are rather high (above 35%) (Hayes et al. 2006a, b). The f2 measure of all three methods exceeds 0.5. It is also worth noting that TF-IDF was able to achieve the higher f2 value at a much higher threshold value (0.2 versus 0.04 for the naïve method).

CM-1 Dataset Next, we examine the results for the CM-1 dataset (See Appendix B, Tables 11, 12, 13 and 14 and Tables 4 and 5). The highest f2 value, 0.315, for the naïve method is at the threshold value of 0.09, with the recall value of 0.525 and precision of 0.12.

The TF-IDF method has the highest f2 value of all methods, 0.528, at a threshold of 0.2 with recall at 0.742 and precision of 0.245. The highest f2 value, 0.43, for the rule-based method is for Rule Family 1 (d_adverbphrase and r_adverbphrase) with the recall value of 0.437 and precision of 0.418. The combination method has its highest f2 value, 0.483, for Rule Families 1, 2, and 4 with recall at 0.586 and precision of 0.356.

Note that the recall is much higher for TF-IDF than for the naïve method (0.742 as compared to 0.525). The rule-based method has unacceptably low recall of less than 1% with one exception (43.7% for two Rule Family 1 rules). The combination method boosts recall to over 62% (0.622) for Rule Family 4.

The precision is much higher for TF-IDF than for naive at 0.245 as compared to 0.12. The rule-based method has unacceptably low precision (ranging from 0 to 0.034) with the exception of two Rule Family 1 rules with precision of 0.418. The combination method achieves much better precision in general, though the highest value is 0.356 (lower than for rule-based). Thus, on the CM-1 dataset, TF-IDF clearly outperformed the naïve method and all other methods.

It should be noted that the best recall value achieved by the naïve method for CM-1 did outperform the recall for TF-IDF, as shown in Table 4. At threshold of 0.03, the naïve method had recall of 0.781 (a little bit better than TF-IDF), but this high recall came with a very low precision of 0.063 (almost four times worse than for the best TF-IDF case), and yielded a value of just 0.238 for f2. Note that the f2 value is much lower than the best value achieved for the naïve method (0.315).

3.5.2 Results for Requirement-Level Satisfaction Labels

As can be seen, in Table 5 and Tables 15, 16, 17, 18 and 19 in Appendix B, the tool generated labels for the Gantt dataset that possessed perfect precision. The TF-IDF and Combination methods had good to excellent probability of detection (pd) (probability of correct classification of a requirement as satisfied) and accuracy (0.588 and 0.764, respectively). All four methods had perfect probability of false alarms (pf) (undefined because there were no false positives and hence divide by zero error), the experts felt that all items deemed satisfied by the tool were truly satisfied, and that all items deemed not satisfied by the tool were indeed not satisfied. This was not the case for the CM-1 dataset, however, as can be seen in Table 5 and Tables 19, 20, 21, 22 and 23 in Appendix B. It is clear that pd varied greatly, ranging from a very poor 0.015 for Rule-based to decent results for TF-IDF and Combination (0.554 and 0.600, respectively). Precision was impressive, with a value of 1.0 for rule-based and with all other methods above 0.886 (it should be noted though that the Rule-based method only deemed one requirement to be satisfied, so precision of 1.0 is not so remarkable - the other methods deemed anywhere from 25 to 39 requirements to be satisfied, so those precision values are meaningful). Accuracy was in the 0.7NN range, with the exception of Rule-based, which was 0.536. Classification accuracy of 70% or higher is generally considered "good." Perhaps the more interesting result for CM-1 was the agreement of the reviewers that many requirements that were labeled as not satisfied were indeed satisfied. This means that there were false negatives. This result provides a clear indication for future improvements.

It should be noted that we were advised[4] to build multiple answer sets for the requirement-level study. In addition to the NASA expert answer sets described above, we also built "majority rules" answer sets for Gantt and CM-1 (for Gantt, this was identical to the NASA expert answer set). We added every item for which there were three opinions to

[4] We consulted with a statistics professor during the answer set creation process.

Table 4 Summary of the chunk-level evaluation

Dataset:	Gantt				CM-1			
	Threshold	Recall	Precision	f-2	Threshold	Recall	Precision	f-2
Naive	0.04	0.67	0.41	0.539	0.09	0.525	0.12	0.315
					0.03	0.781	0.063	0.238
TF-IDF	0.2	0.664	0.486	0.619	0.2	0.742	0.245	0.528
Rules-based	N/A, Rule family 1 – RF1, d_adverb-phrase, r_adverb-phrase	0.198	0.162	0.185	N/A, RF1, d_adverb-phrase, r_adverb-phrase	0.437	0.418	0.430
Combination	N/A, RF2, d_adverb-phrase	0.73	0.499	0.632	N/A, RF1,RF2,RF4, r_adverbphrase,d_adjective-phrase, r_adjective-phrase, d_interjection, r_inter-jection, d_particle	0.586	0.356	0.483

Table 5 Summary of the requirement-level evaluation

Dataset:	Gantt				CM-1			
	pd	Pf	Precision	Accuracy	pd	Pf	Precision	Accuracy
Naïve 0.04	0.176	**undef**ined	**1**	0.176	NA	NA	**NA**	NA
Naïve 0.03	NA	NA	NA	NA	0.492	0.055	0.889	0.732
Naïve 0.09	NA	NA	NA	NA	0.385	0.014	0.962	0.703
TF-IDF	0.588	**Undef**	**1**	0.588	0.554	0.055	**0.900**	0.761
Rules-based	0.176	**Undef**	**1**	0.176	0.015	**0.00**	**1.00(*)**	0.536
Combination	**0.764**	**Undef**	**1**	**0.764**	**0.600**	**0.068**	0.886	**0.775**

*only one requirement was deemed satisfied; **bolded items** were the "best" for that measure

the answer set (using the majority opinion as the label). The resulting confusion matrices were quite similar to those presented in the Results section. For example, the TF-IDF results were: pd 0.617, precision 0.875, pf 0.125, and accuracy of 0.724 as compared to the NASA expert answer set which yielded pd 0.554, precision 0.900, pf 0.055, and accuracy of 0.761. As the results were very close, here we present only the NASA expert answer set results.

As a number of reviewers were used in performing the requirement-level validation, we examined inter rater agreement for each dataset. We used free marginal multi rater kappa for inter rater agreement, as our reviewers were not "forced' to provide opinions on every item given to them (and indeed they did choose to skip some items) (Kappa). For the Gannt dataset, there were 17 requirements examined by all three reviewers with two categories (satisfied or not satisfied), with a kappa of 0.92. For CM-1, there were 58 requirements for which all three reviewers provided opinions, with a kappa of 0.540. Values of 0.4 to 0.6 are considered to be in moderate agreement (CM-1), the Gantt result would be considered a very good level of agreement (Altman 1991).

3.5.3 Comparison of Chunk-Level Satisfaction Assessments and Requirement-Level Satisfaction Labels

As shown in Table 4, at the chunk level, for the Gantt data set, the Combination method provided the highest level of precision (0.499) and also the highest recall (0.73). For the CM-1 data set at the chunk level, the TF-IDF method had the highest recall (0.742), and the Rules-based method had the highest precision (0.418). At the requirement level, each method performed equally well on the Gantt data set with regard to precision (1), and the Combination method provided the greatest accuracy (0.764), as shown in Table 5. For the CM-1 data set at the requirement level, the Rules-based method had the highest precision (1), but did not return multiple labels. The Naive method at the 0.09 threshold had the next best precision (0.962). At the requirement level, the best accuracy for CM-1 was found with the Combination method (0.775). Overall, we would recommend the Combination method. Due to increased recall and precision at the chunk level, the overall accuracy of the results returned at the requirement level is increased. Also, in the requirement-level study, the Combination method had accuracy that was 29.9% better than the next best (TF-IDF) method for Gantt, and also outperformed the accuracy of other methods by a smaller margin on the CM-1 data set. By increasing the accuracy of results, analyst workload is reduced. It is a much faster exercise to verify the results of a given requirement and corresponding satisfying design elements than to search the entire design space for those design elements that satisfy the requirement.

4 Analysis and Discussion

In general software engineering applications of information retrieval methods, it has been found that recall is higher and precision is lower for larger datasets (such as CM-1) when applying the naive and TF-IDF satisfaction assessment methods.

The *f2* measure exceeds 0.5 for the TF-IDF method, but not for the naïve method. These methods return results based on textual similarity. With a larger dataset, it is more likely that textual patterns will emerge and similar terms will be used in multiple locations. Larger datasets tend to have better selectivity for the TF-IDF and rules-based satisfaction assessment methods. This is due to the weighting of potential matches based on term frequency and inverse document frequency. Terms that occur frequently in large datasets contribute less in determining satisfaction mappings, whereas in smaller datasets the frequency that a term appears in the overall dataset may not be filtered out due to small document size. Additionally, the content of data sets impacts the recall and precision of each method. In this study, the CM-1 data set contained a higher density of unique keywords, such as acronyms and domain-specific terms, than were contained in the Gantt data set. These unique keywords are more likely to generate a satisfaction mapping between two chunks, but may also lead to more false positives than in data sets that contain more generic language. In data sets with more generic language, it is the cumulative effect of multiple matching keywords that generate a candidate satisfaction mapping, whereas a unique term in both a requirement and design element chunk alone could generate a candidate satisfaction mapping in a data set with a moderate to high density of unique keywords. Since unique keywords are more similar textually, this may have also contributed to the performance of the naïve and TF-IDF satisfaction assessment methods on CM-1.

For the rules-based and combination methods at the chunk level, recall is higher on average for the Gantt dataset than for the CM-1 dataset. The Gantt dataset uses fewer domain-specific terms and these terms are more likely to be found by the rule-based approach than many of the terms found in the CM-1 dataset. At the requirement level, accuracy was higher for every method for the CM-1 dataset results than for the Gannt data set results. This, again, is likely a factor of the density of domain-specific terms within CM-1.

Looking at method performance, the naïve satisfaction assessment method captured links based solely on textual similarity. This yielded a lower level of precision at the chunk level than the other methods. This trend was observed at the requirement level as well. The recall for the naïve satisfaction assessment method was slightly higher than other methods, but this was due to the large number of links (often false) being retained (thus forcing an analyst to look through far more candidate links). The TF-IDF satisfaction assessment method showed reasonable performance overall, with strong recall and precision for both datasets at the chunk level. Likewise, TF-IDF performed well at the requirement level. While no single rule set has the recall and precision values of TF-IDF in the chunk-level study, the rules-based satisfaction assessment method captured candidate satisfaction mappings that were not caught by other satisfaction assessment methods. Finally, the combination satisfaction assessment method at the chunk level achieved a higher recall than the TF-IDF method alone with a reasonable tradeoff in precision for the Gantt data set and a high level of recall and precision for the CM-1 data set. In the requirement level study, the accuracy of the Combination method was best for both Gantt and CM-1 data sets.

In a perfect world, a tool would be able to return results with both perfect recall and perfect precision. For satisfaction assessment, we would like to be able to capture links that are both textually similar and semantically similar. If we choose to value recall over precision, then we increase the work of the analyst, who must sift through and eliminate

false positives. If we choose to value precision over recall, we will return some satisfaction mappings, but the analyst must search through the full set to gather additional satisfaction mappings. Those satisfaction mappings that are not returned by the tool are strong indicators that ambiguous language has been used or that requirements are weakly satisfied. Discovering characteristics and ways to identify weakly satisfied requirements is an area of future work.

5 Related Work

Satisfaction assessment may be thought of as a way to validate whether requirements have been fully addressed by design elements and provides a way to measure the quality of a software project. The majority of previous automation work in the requirements traceability community focused on candidate link generation, not on satisfaction assessment. This work, however, has paved the way for our study and is briefly described here.

Previous work on requirements validation has focused on formally specifying requirements (Spivey 1988; Robinson and Pawlowski 1999; Greenspan et al. 1994), optimizing natural language processing (NLP) approaches to requirements analysis, and discovering potential ambiguities (Lecceuche 2000). We use NLP in the rule-based method and build our satisfaction assessment tool upon prior work in NLP parsing, tagging, and chunking.

Durán et al. used XSLT and requirements in XML to automatically verify requirement qualities such as completeness and lack of ambiguity (Durán et al. 2001). Their work, unlike the work presented in this paper, involved modifying existing requirements and design to fit a specified format. Satisfaction assessment as presented here can help address the question of completeness given requirements and design. Analysts have used requirement defect detection techniques (Porter and Votta 1998) to discover requirements that cannot be satisfied (i.e., inconsistent and omitted) and inconsistencies between requirements and design. Their work can be used to find potential defects, but does not examine coverage or satisfaction of requirements by design. Reading methods such as scenario-based (Sutcliffe 1998) and perspective-based reading (Letier and van Lamsweerde 2004; Shull et al. 2000) have also been used to increase the quality of requirement specifications. Their techniques are used to discover a subset of defects in requirements documents, but do not explicitly examine coverage or satisfaction. Our work seeks to improve requirement specification/design document pairs. If requirements are detected that are not fully satisfied by the design elements, it is possible that the requirement was not properly specified and requires correction.

Automation of requirements tracing has received extensive attention in recent years. For a detailed survey, we refer the reader to prior work (Hayes et al. 2006a, b). Tracing examines the creation of a requirements traceability matrix (RTM) that relates requirements to design to code and beyond. Recent work concentrated on applying IR methods to tracing. Antoniol (2002) and Marcus and Maletic (2003) applied IR methods to the problem of tracing design to code. Cleland-Huang et al. (2005) used IR to trace non-functional requirements. Hayes et al. investigated the process of tracing and built a special-purpose requirements tracing tool called REquirements TRacing On-target (RETRO) (Hayes et al. 2003, 2006a, b). Spanoudakis et al. created a system to automatically generate traceability information based on tracing rules (Spanoudakis et al. 2003). We use IR methods to determine which design element chunks map to requirement element chunks. For example, we use TF-IDF in some methods to determine the similarity measures of individual requirement and design element chunks. This work differs from prior tracing in that it looks at requirements satisfaction rather than tracing (Antoniol 2002;

Marcus and Maletic 2003; Spanoudakis and Zisman 2005). That is, it looks at which words or phrases within a design element contribute to the satisfaction of particular subportions of a requirement. Additionally, new methods for similarity comparison including a rule definition language and combination methods using both traditional IR methods and the natural language rule system have been introduced.

Additionally, several researchers have examined requirement quality through design and requirement analysis. Diallo et al. used ScenarioML to create mappings between requirement-level scenarios and system architecture (2007). Their work required formatting requirements and design elements as input into the ScenarioML tool and looked at tracing-level qualities of relatedness. Alspaugh and Antón examined automation of requirement scenario analysis to determine requirement quality, looking at four primary traits: well-definedness, coverage, minimality, and coherence (2008). Their work highlighted specific defect categories, looking for omissions and errors rather than examining satisfaction between two document sets. Robinson looked at rule-based requirements monitors to dynamically analyze requirements as a system is designed (2005). Robinson also expressed user expectations as goals and developed methods to see if the goals were met (2009). Each technique involved analyst input in creating appropriate monitors and goals rather than free-form requirements. Robinson used a variant of Object Constraint Language- Temporal Message logic (OCL-TM) to automatically acquire user satisfaction feedback (2009). In contrast, we examine requirements to see if they are satisfied by design elements. Our work does not require formal specification. Our method is static. Letier and van Lamsweerde created a system to analyze partial goal satisfaction to help quantify the impact of partially met requirements due to design constraints (2004).

Algorithmic techniques that are useful for both assessing requirement satisfaction and tracing include keyword extraction methods (Hayes et al. 2006a, b) and the vector space model for information retrieval (Salton 1983). Vector space models represent documents as vectors by extracting terms, weighting these by relevance and document location, and ranking the document as a whole based on a given query. We apply the vector space model in our TF-IDF and Combination methods.

6 Conclusions and Future Work

Automated techniques have been implemented to assist analysts in assessing which design elements satisfy requirements, reducing the time and effort required for verification. Four satisfaction assessment methods have been proposed and evaluated: Naïve satisfaction assessment, TF-IDF satisfaction assessment, rule-based satisfaction assessment, and Combination satisfaction assessment.

It was found that the TF-IDF satisfaction assessment method had higher precision than the naïve satisfaction assessment method. Using the TF-IDF satisfaction assessment method instead of the naïve satisfaction assessment method will result in a candidate satisfaction assessment that requires less analyst effort to verify. The rule-based satisfaction assessment method has shown promise in discovering candidate satisfaction mappings that are not found by other methods. The Combination satisfaction assessment method, which uses both TF-IDF and rule-based satisfaction assessment approaches, has a higher level of recall than either method alone and can further reduce analyst time required for satisfaction assessment verification.

Trends in data sets were also described. In general, when satisfaction assessment is performed on larger data sets, the resulting candidate satisfaction assessments will have higher recall and lower precision. As can be seen, three of the four methods performed well

on the Gantt dataset with recall values above 66%, precision values above 40%, and f2 values above 0.5. The TF-IDF method showed acceptable performance on the CM-1 dataset, with the other three methods falling a bit short.

We believe that we have successfully established that the naïve, TF-IDF, and combination methods can serve as viable baseline methods for assessing performance of automated satisfaction assessment techniques. The naïve method, in particular, is very simple and delivers decent performance with relatively little effort expended.

To a large degree, the performance of the naïve method establishes a measuring stick for us. Any automated methods that fall short of its performance shall be deemed unacceptable for dealing with satisfaction assessment. Methods that exhibit similar performance shall be considered candidates for further improvement; but if no further improvement is achieved, Occam's razor dictates that we abandon them in favor of the naïve method as well.

We believe there are a number of areas for future work. As a long term goal, we would like to evaluate additional datasets as well as establish methods to automatically determine the optimal threshold values for each method. Additional data set studies could also yield insights on whether RTM density, unique keyword density, and other dataset qualities are correlated with satisfaction levels. Optimal threshold values will likely vary by dataset size (number of requirements, design elements, and RTM density) and domain. In addition, future studies include developing measures of requirement quality. The two datasets we used in this work are but a drop in the ocean of possible types of software engineering artifacts that may undergo satisfaction assessment. Because of this, we are not in a position to draw any strong general conclusions about the applicability of specific methods to different types of documents. The only observation we feel comfortable making in this respect is that when requirements are written using proper grammar, punctuation, spelling, etc., tagging and natural language processing algorithms often perform better. It would be very interesting to examine whether the quality of writing within requirement and design elements has an impact on the results of this satisfaction assessment and other information retrieval techniques. One could examine this based on quality of grammar, readability (i.e. Flesch-Kinkaid), or a variety of other metrics. It would be interesting to examine automated thesaurus generation, ways to recognize and map domain-specific terminology, and other ways to generate inputs to the tool (ontologies of terms, acronym lists, RTMs, etc.) in an automated fashion.

We hope to identify ways to identify weakly satisfied requirements. We hope to develop additional satisfaction assessment methods. We plan to further examine the thresholds used. We plan to examine potential new methods, including an approach that generates chunk matches based on a set of predefined rules on content and grammatical structure of the chunks. We plan to examine the large number of false negatives detected by the evaluators in the CM-1 dataset in order to identify ways to improve our methods. In addition, validation needs to be expanded. We plan to apply our work to additional datasets in other domains. We also plan to examine the applicability of our methods to different artifact pairs. A further study of the analyst, including analysis of time spent performing manual versus automation-assisted satisfaction assessment and perceived effort in the manual process versus the automated process is also an area of future research.

Acknowledgments This work is funded in part by the National Science Foundation under NSF grant CCF-0811140. This work was partially sponsored by NASA under grant NNG05GQ58G. We thank David Pruett and the other evaluators for their help. We thank Hakim Sultanov and Bill Kidwell. Thanks to Stephanie Ferguson, Marcus Fisher, Ken McGill, Tim Menzies, Lisa Montgomery, and everyone at the NASA IV&V facility. Thanks also to fellow graduate students Jody Larsen, Senthil Sundaram, Liming Zhao, and Sravanthi Vadlamudi. We thank statistics professor Dr. Arnold Stromberg.

Appendix A: Dataset Examples

Below are examples from each of the datasets used in this paper.

CM-1

Requirements:

SRS5.12.2.2. The DPU-CCM shall process real-time non-deferred commands within B ms of receipt from the ICU or the SCU.
SRS5.13.1.1. The DPU-TMALI shall install callbacks for handling all DPU-DCI interrupts including Error interrupt, Ping-Pong Timeout interrupt, and Ping-Pong Complete Flag interrupt.
SRS5.2.3.1. The DPU-RTOS shall exclude failed DRAM from the system memory pool based on the contents of the BIT_DRAM results in the SYSTEM_BLOCK. The system memory table does not include the Interrupt Vector Table (IVT), nor the text and data segment.

Design Elements:

DPUSDS5.2.3.6.1. Install Exception Handlers In the diagnostic mode of operation, the RSC processor generates external interrupts for memory single-bit errors (SBEs), multiple-bit errors (MBEs), and address exceptions. The RSCVME Board Support Package of VxWorks? does not directly support access to these interrupts. Some custom routines must be provided to access the Memory Error Interrupt.
DPUSDS5.12.1.4.1. Memory Upload and Download Handling There are two ways to upload data to the DPU:* Memory Poke (D_MEM_DAT_POKE command), or* Memory Upload (D_MEM_DAT_UPLD command).The memory poke command is used when a small (<=Z bytes) of data need to be poked into a DPU memory location. The Z byte limitation is derived from the Company X command length constraint.
DPUSDS5.2.3.6.5. Install Exception Handlers The RSC processor also generates an external interrupt for the Power Fail Interrupt. The RSCVME Board Support Package of VxWorks? does not directly support access to this interrupt. Some custom routines must be provided to access this interrupt. These functions are described below, and are contained in sysLibSup.c.

Gantt

Requirements:

R6. Create Resources (person); GanttProject supports Persons as resources. Persons have names and holidays or vacation days. Persons can be assigned to work on tasks.
R12. Change Task Begin/End Times automatically with dependency changes; The start or end date should be changed automatically if links among tasks are changed
R16. Add/Remove Holidays and Vacation Days; Holidays and vacation days are properties of persons (resources). changing this information also changes the availability of a person on certain days.

Design Elements:

DE4-1. To add tasks as subtasks a method which indent the selected task nodes in GUI and change them to be subtasks is used. A manager of task hierarchy provides functions to update the relationship between tasks.

DE10-4. The human resource class can have multiple objects of resource assignments which assigns this resource to tasks. The class provides function to get the list of these objects.
DE14-1. A GUI class of graphic area provides a function to draw dependency. The function uses an object of the task manager to add dependencies.

Appendix B: Additional Tables

Table 6 Number of possible k-element rule sets

K	Number of Possible Rule Sets
1	242,000,000
2	2.9281999879e+16
3	2.36208130405133e+24
4	1.42905917123545e+32
5	6.91664627445483e+39
6	2.78971393972473e+47
7	9.64443938107287e+54
8	2.9174428283857e+62
9	7.84467934588663e+69
10	1.89841233110245e+77

Table 7 Gantt dataset naïve method results

Filter Value	Naïve - Overall Recall	Naïve - Overall Precision	Naïve - F2-Measure
0.01	0.678	0.294	0.537
0.02	0.678	0.294	0.537
0.03	0.678	0.295	0.538
0.04	0.674	0.299	0.539
0.05	0.645	0.312	0.531
0.06	0.635	0.312	0.526
0.07	0.557	0.325	0.487
0.08	0.554	0.325	0.485
0.09	0.463	0.36	0.438
0.1	0.433	0.363	0.417
0.2	0.078	0.32	0.092
0.3	0	0	0

Table 8 Gantt dataset TF-IDF method results

Filter Value	TF-IDF - Overall Recall	TF-IDF - Overall Precision	TF-IDF - F2-Measure
0.01	0.664	0.486	0.619
0.02	0.664	0.485	0.619
0.03	0.664	0.485	0.619
0.04	0.664	0.485	0.619
0.05	0.664	0.485	0.619

Table 8 (continued)

Filter Value	TF-IDF - Overall Recall	TF-IDF - Overall Precision	TF-IDF - F2-Measure
0.06	0.664	0.485	0.619
0.07	0.664	0.485	0.619
0.08	0.664	0.485	0.619
0.09	0.664	0.486	0.619
0.1	0.664	0.486	0.619
0.2	0.664	0.486	0.619
0.3	0.58	0.492	0.56
0.4	0.554	0.528	0.548
0.5	0.521	0.565	0.529
0.6	0.423	0.647	0.455
0.7	0.391	0.764	0.433
0.8	0.339	0.819	0.384
0.9	0.313	0.835	0.357

Table 9 Gantt dataset rule-based method results

Rule Family-Rule	Rule-based - Overall Recall	Rule-based - Overall Precision	Rule-based-F2-Measure
RF1-d_nounphrase	0.029	0.092	0.0383
RF1-d_verbphrase	0.198	0.162	0.185
RF1-r_nounphrase	0.029	0.092	0.038
RF1-r_verbphrase	0.198	0.162	0.185
RF2-d_adjectivephrase	0.042	0.304	0.059
RF2-d_adverbphrase	0.004	0.5	0.0059
RF2-r_adjectivephrase	0.042	0.304	0.059
RF2-r_adverbphrase	0.004	0.5	0.0059
RF3-d_conjunction	0.138	0.31	0.169
RF3-d_list	0	0	N/A
RF3-r_conjunction	0.138	0.31	0.169
RF3-r_list	0	0	N/A
RF4-d_interjection	0	0	N/A
RF4-d_particle	0	0	N/A
RF4-d_preposition	0.0011	0.0016	0.0013
RF4-r_interjection	0	0	N/A
RF4-r_particle	0	0	0.0013
RF4-r_preposition	0.0011	0.0016	N/A

Table 10 Gantt dataset combination of rule-based and TF-IDF method results

Rule Family-Rule	Combination- Overall Recall	Combination - Overall Precision	Combination-F2-Measure
RF1-d_nounphrase	0.673	0.535	0.62
RF1-d_verbphrase	0.667	0.567	0.63
RF1-r_nounphrase	0.691	0.477	0.60
RF1-r_verbphrase	0.667	0.569	0.63
RF2-d_adjectivephrase	0.667	0.569	0.63
RF2-d_adverbphrase	0.729	0.49	0.632
RF2-r_adjectivephrase	0.667	0.569	0.631
RF2-r_adverbphrase	0.667	0.519	0.609
RF3-d_conjunction	0.673	0.508	0.607
RF3-d_list	0.673	0.535	0.620
RF3-r_conjunction	0.667	0.567	0.630
RF3-r_list	0.691	0.477	0.601
RF4-d_interjection	0.667	0.569	0.631
RF4-d_particle	0.667	0.569	0.631
RF4-d_preposition	0.729	0.49	0.631
RF4-r_interjection	0.667	0.569	0.631
RF4-r_particle	0.667	0.519	0.609
RF4-r_preposition	0.673	0.508	0.607

Table 11 CM-1 dataset naïve method results

Filter Value	Naïve - Overall Recall	Naïve - Overall Precision	Naïve - F2-Measure
0.01	0.783	0.063	0.237
0.02	0.783	0.063	0.237
0.03	0.781	0.063	0.238
0.04	0.759	0.064	0.239
0.05	0.69	0.068	0.245
0.06	0.681	0.07	0.248
0.07	0.635	0.087	0.281
0.08	0.611	0.085	0.273
0.09	0.525	0.121	0.315
0.1	0.449	0.113	0.282
0.2	0.1	0.075	0.094
0.3	0	0	0

Table 12 CM-1 dataset TF-IDF method results

Filter Value	TF-IDF - Overall Recall	TF-IDF - Overall Precision	TF-IDF - F2-Measure
0.01	0.746	0.235	0.52
0.02	0.746	0.235	0.52
0.03	0.746	0.235	0.52
0.04	0.746	0.235	0.52
0.05	0.746	0.235	0.52
0.06	0.746	0.235	0.52
0.07	0.746	0.235	0.52
0.08	0.746	0.235	0.52
0.09	0.746	0.235	0.52
0.1	0.746	0.235	0.52
0.2	0.742	0.245	0.528
0.3	0.694	0.264	0.524
0.4	0.606	0.287	0.495
0.5	0.531	0.309	0.464
0.6	0.445	0.328	0.416
0.7	0.368	0.36	0.366
0.8	0.307	0.376	0.318
0.9	0.26	0.352	0.274

Table 13 CM-1 dataset rule-based method results

Rule Family-Rule	Rule-based - Overall Recall	Rule-based - Overall Precision	Rule-based-F2-Measure
RF1-d_nounphrase	0.026	0.135	0.035
RF1-d_verbphrase	0.437	0.418	0.43
RF1-r_nounphrase	0.026	0.135	0.035
RF1-r_verbphrase	0.437	0.418	0.43
RF2-d_adjectivephrase	0.0344	0.121	0.0452
RF2-d_adverbphrase	0.0017	0.0191	0.0025
RF2-r_adjectivephrase	0.034	0.121	0.0452
RF2-r_adverbphrase	0.0017	0.0191	0.0025
RF3-d_conjunction	0.064	0.074	0.067
RF3-d_list	0	0	N/A
RF3-r_conjunction	0.064	0.074	0.067
RF3-r_list	0	0	N/A
RF4-d_interjection	0	0	N/A
RF4-d_particle	0	0	N/A
RF4-d_preposition	0.0013	0.0012	0.0013
RF4-r_interjection	0	0	N/A
RF4-r_particle	0	0	N/A
RF4-r_preposition	0.0013	0.0012	0.0013

Table 14 CM-1 dataset combination of rule-based and TF-IDF method results

Rule Family-Rule	Combination- Overall Recall	Combination - Overall Precision	Combination-F2-Measure
RF1-d_nounphrase	0.598	0.31	0.457
RF1-d_verbphrase	0.586	0.337	0.47
RF1-r_nounphrase	0.613	0.243	0.407
RF1-r_verbphrase	0.586	0.356	0.483
RF2-d_adjectivephrase	0.586	0.356	0.483
RF2-d_adverbphrase	0.622	0.216	0.383
RF2-r_adjectivephrase	0.586	0.356	0.483
RF2-r_adverbphrase	0.587	0.248	0.403
RF3-d_conjunction	0.588	0.298	0.444
RF3-d_list	0.59	0.310	0.457
RF3-r_conjunction	0.586	0.337	0.470
RF3-r_list	0.613	0.243	0.407
RF4-d_interjection	0.586	0.356	0.483
RF4-d_particle	0.586	0.356	0.483
RF4-d_preposition	0.622	0.216	0.383
RF4-r_interjection	0.586	0.356	0.483
RF4-r_particle	0.587	0.248	0.403
RF4-r_preposition	0.588	0.298	0.444

Table 15 Gantt results for naïve method

Naive 0.04		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	3	14
	Not Satisfied	0	0
pd	0.176	precision	1
pf	undefined	accuracy	0.176

Table 16 Gantt results for TF-IDF method

TF-IDF		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	10	7
	Not Satisfied	0	0
pd	0.588	precision	1
pf	undefined	accuracy	0.588

Table 17 Gantt results for rule-based method

Rule-Based		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	3	14
	Not Satisfied	0	0
pd	0.176	precision	1
pf	undefined	accuracy	0.176

Table 18 Gantt results for combination method

Combination		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	13	4
	Not Satisfied	0	0
pd	0.764	precision	1
pf	undefined	accuracy	0.764

Table 19 CM-1 results for naive method, 0.03 threshold

Naive 0.03		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	32	33
	Not Satisfied	4	69
pd	0.492	precision	0.889
pf	0.055	accuracy	0.732

Table 20 CM-1 Results for naive method, 0.09 threshold

Naive 0.09		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	25	40
	Not Satisfied	1	72
pd	0.385	precision	0.962
pf	0.014	accuracy	0.703

Table 21 CM-1 results for TF-IDF method

TF-IDF		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	36	29
	Not Satisfied	4	69
pd	0.554	precision	0.900
pf	0.055	accuracy	0.761

Table 22 CM-1 results for rule-based method

Rule-Based		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	1	64
	Not Satisfied	0	73
pd	0.015	precision	1.000
pf	0.000	accuracy	0.536

Table 23 CM-1 Results for combination method

Combination		Tool Generated Label	
		Satisfied	Not Satisfied
Actual	Satisfied	39	26
	Not Satisfied	5	68
pd	0.600	precision	0.886
pf	0.068	accuracy	0.775

References

Altman DG (1991) Practical statistics for medical research. Chapman & Hall

Antoniol G (2002) Recovering traceability links between code and documentation. IEEE Trans on Software Engineering 28(10):970–983

Asplaugh TA, Antón AI (2008) Scenario support for effective requirements. Inf Softw Technol 50 (3):198–220

Baeza-Yates R, Ribeiro-Neto B (2003) Modern information retrieval. Addison-Wesley

Cleland-Huang J (2002) Automating speculative queries through event-based requirements traceability. Joint Conference on Requirements Engineering

Cleland-Huang J, Settimi R, BenKhadra O, Berezhanskaya E, Christina S (2005) Goal-centric traceability for managing non-functional requirements. In: Proceedings of the 27th international conference on Software engineering (ICSE '05). ACM, New York, NY, pp 362–371. doi:10.1145/1062455.1062525. http://doi.acm.org/10.1145/1062455.1062525

Cuddeback D, Dekhtyar A, Hayes JH (2010) Automated requirements traceability: the study of human analysts. In Proc. 18th International Conference on Requirements Engineering, Sydney, Australia

Di Lucca GA, Di Penta M, Antoniol G, Casazza G (2001) An approach for reverse engineering of web-based applications. BP - 231, Dipartimento di Informatica e Sistemistica

Diallo MH, Naslavsky L, Ziv H, Alspaugh TA, Richardson DA (2007) Evaluating software architectures against requirements-level scenarios. Workshop on the Role of Software Architecture for Testing and Analysis

Durán A, Ruiz A, Toro M (2001) An automated approach for verification of software requirements. Jornadas de Ingeniería de Requisitos Aplicada, Seville, Spain

GanttProject, http://ganttproject.biz/

Gotel OCZ, Finkelstein ACW (1996) Extended requirements traceability: a framework for changing requirements. Workshop on Requirements Engineering in a Changing World

Greenspan S, Mylopoulos J, Borgida J (1994) On formal requirements modeling languages: RML revisited. Proc. 16th International Conference on Software Engineering, p 135–147, Sorrento, Italy

Hayes JH, Dekhtyar A (2005) Humans in the traceability loop: can't live with 'em, can't live without 'em. Proc. of the 3rd International Workshop on Traceability in Emerging Forms of Software Engineering, Long Beach, California. TEFSE '05. ACM, New York, NY, 20–23

Hayes JH, Dekhtyar A, Osbourne J (2003) Improving requirements tracing via information retrieval. International Conference on Requirements Engineering

Hayes JH, Dekhtyar A, Sundaram S (2006a) Advancing requirements tracing: the study of methods. IEEE Trans Software Engineering 32(1):4–19

Hayes JH, Dekhtyar A, Sundaram S (2006) Advances in dynamic generation of traceability links. Tech Report, (TR 451–06)

Hayes JH, Dekhtyar A, Sundaram S, Holbrook A, Vadlamudi S, April A (2007) REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery. Innovations in Systems and Software Engineering: A NASA Journal 3(3):193–202

Holbrook EA (2009) Satisfaction assessment of textual software engineering artifacts, PhD dissertation, Dept. of Computer Science., University of Kentucky, Lexington, KY

Holbrook EA, Hayes JH, Dekhtyar A (2009) Toward automating requirements satisfaction assessment. Requirements Engineering Conference, 2009. RE '09. 17th IEEE International, vol., no., pp 149–158

ISO9000:2000, Quality Standard, International Organization for Standardization

Lecceuche R (2000) Finding comparatively important concepts between texts. Automated Software Engineering (ASE'00). Washington, DC, 55

Letier E, van Lamsweerde A (2004) Reasoning about partial goal satisfaction for requirements and design engineering. SIGSOFT Softw Eng Notes 29(6):53–62

Marcus A, Maletic JI (2003) Recovering documentation-to-source code traceability links using latent semantic indexing. In: Proceedings of the 25th International Conference on Software Engineering (ICSE '03). IEEE Computer Society, Washington, DC, pp 125–135

Marcus MP, Santorini B, Marcinkiewicz MA (1993) Building a large annotated corpus of English: the Penn Treebank. Computational Linguistics 19:313–330

Marcus A, Maletic JI, Sergeyev A (2005) Recovery of traceability links between software documentation and source code. Int J Softw Eng Knowl Eng, World Scientific 15(5):811–836

Porter A, Votta L (1998) Comparing detection methods for software requirements inspections. Empir Softw Eng 3(4):355–379

Robinson WN (2005) Implementing rule-based monitors within a framework for continuous requirements monitoring. In: Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences - Volume 07 (HICSS '05), Vol. 7. IEEE Computer Society, Washington, DC, 188.1-. doi:10.1109/HICSS.2005.306. http://dx.doi.org/10.1109/HICSS.2005.306

Robinson WN (2009) Seeking quality through user-goal monitoring. IEEE Software, pp 58–65

Robinson WN, Pawlowski S (1999) Managing requirements inconsistency with development goal monitors. IEEE Trans. on Software Eng

Salton G (1983) Introduction to modern information retrieval. McGraw-Hill

Shull G, Rus I, Basili VR (2000) How perspective-based reading can improve requirements inspections. IEEE Computer 33(7):73–79

Spanoudakis G, Zisman A (2005) Software traceability: a roadmap. In: Chang SK (ed) Handbook of Software Engineering and Knowledge Engineering, vol. 3: Recent Advancements. World Scientific Publishing

Spanoudakis G, d'Avila A, Garcez A, Zisman A (2003) Revising rules to capture requirements traceability relations: a machine learning approach. In: Proceedings of the 15th International Conference in Software Engineering and Knowledge Engineering (SEKE 2003), pp 570–577, San Francisco

Spanoudakis G, Zisman A, Pérez-Miñana E, Krause P (2004) Rule-based generation of requirements traceability relations. J Syst Softw 2(72):105–127

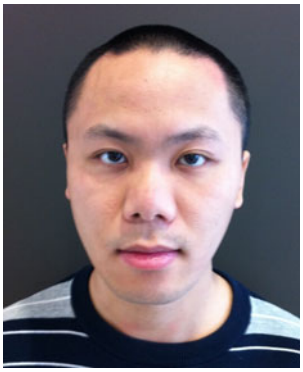
Spivey J (1988) Understanding Z. Cambridge

Sutcliffe A (1998) Scenario-based requirement analysis. Requir Eng 3(1):48–65. doi:10.1007/BF02802920. http://dx.doi.org/10.1007/BF02802920

Elizabeth Ashlee Holbrook received the PhD degree in computer science from the University of Kentucky, where she was a NSF Graduate Research Fellow. She is a the Firmware Test Team Lead for the Security, Integration, and Solutions Framework teams at Lexmark International and is a part-time instructor of Requirements Engineering at the University of Kentucky. Her interests include satisfaction assessment, requirements engineering, traceability, and software quality assurance.

Jane Huffman Hayes received the PhD degree in information technology from George Mason University. She is an associate professor in the Department of Computer Science at the University of Kentucky. Her research interests include requirements, software verification and validation, traceability, maintainability, and reliability. She is the current elected Director of the Center of Excellence for Software Traceability. She is a member of the IEEE Computer Society and the Association for Computing Machinery.

Alex Dekhtyar received the PhD degree in computer science from the University of Maryland at College Park. He is an associate professor in the Department of Computer Science at California Polytechnic State University. His interests include traceability, management and reasoning with uncertain information, digital libraries, computing in humanities, and management of XML data. He is a member of the ACM, ACM SIGMOD, SIGIR, and SIGKDD, American Association for AI, and Association for Logic Programming.



Wenbin Li is currently pursuing the PhD degree at the University of Kentucky, he has completed all graduation requirements except for dissertation.